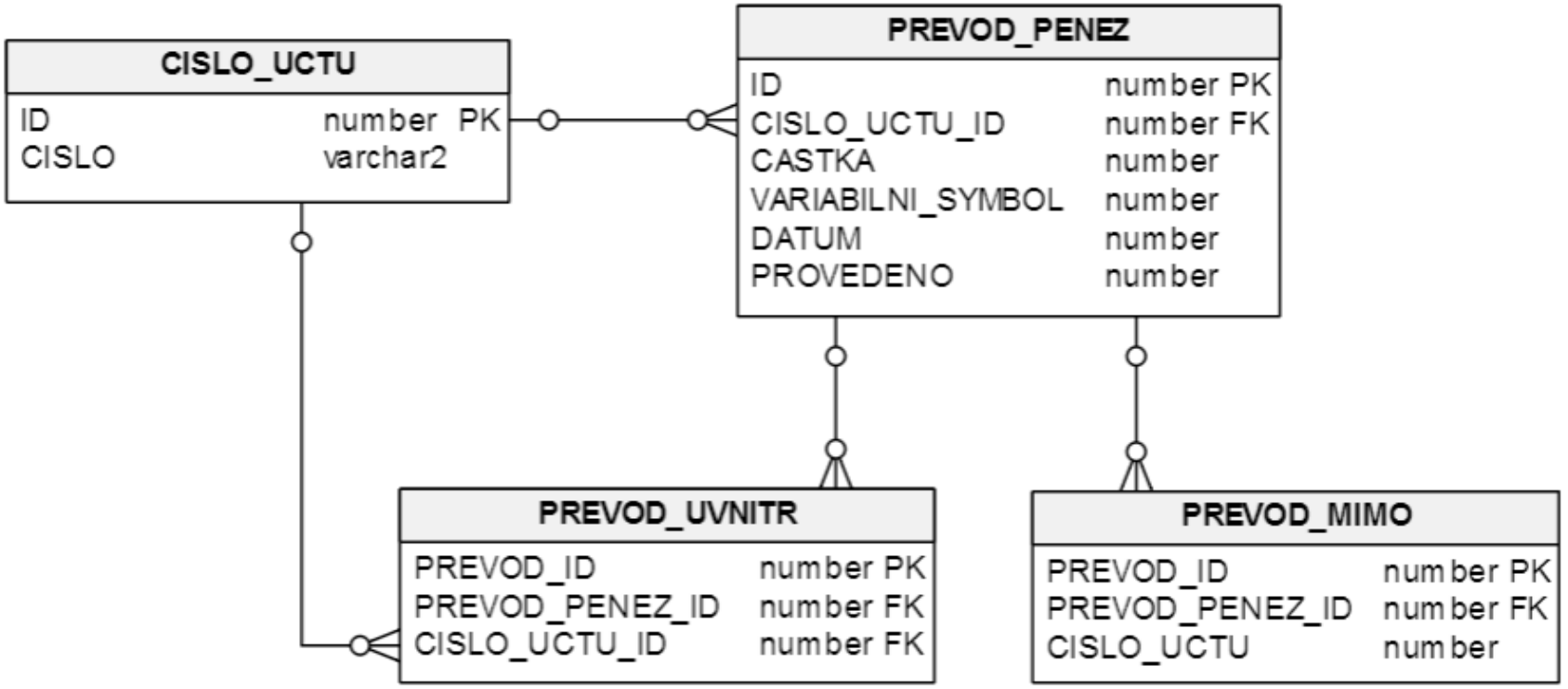
Návrh databáze

Normalizace, best practice

- Návrh tabulek je vždy kompromisem mezi flexibilitou a výkonem jednotlivých transakcí.
- Flexibilní návrh databáze je alespoň ve třetí normální formě.
  - http://cs.wikipedia.org/wiki/T%C5%99et%C3%AD_norm%C3%A1ln%C3%AD_forma
    - Každá tabulka má primární klíč.
    - Každý neklíčový atribut není závislý na ničem jiném než klíči (má úzký vztah ke klíči).
- Drtivá většina databáze by měla být navržena ve třetí normální formě. Tento přístup má ale tu nevýhodu, že při jeho dodržování máte kupu tabulek a vazeb mezi nimi.
- Poté se může jedna transakce skládat z celé řady operací, které snižují výkon databáze. Proto v takových situacích můžete přistoupit k denormalizaci a tím zvýšit výkon.

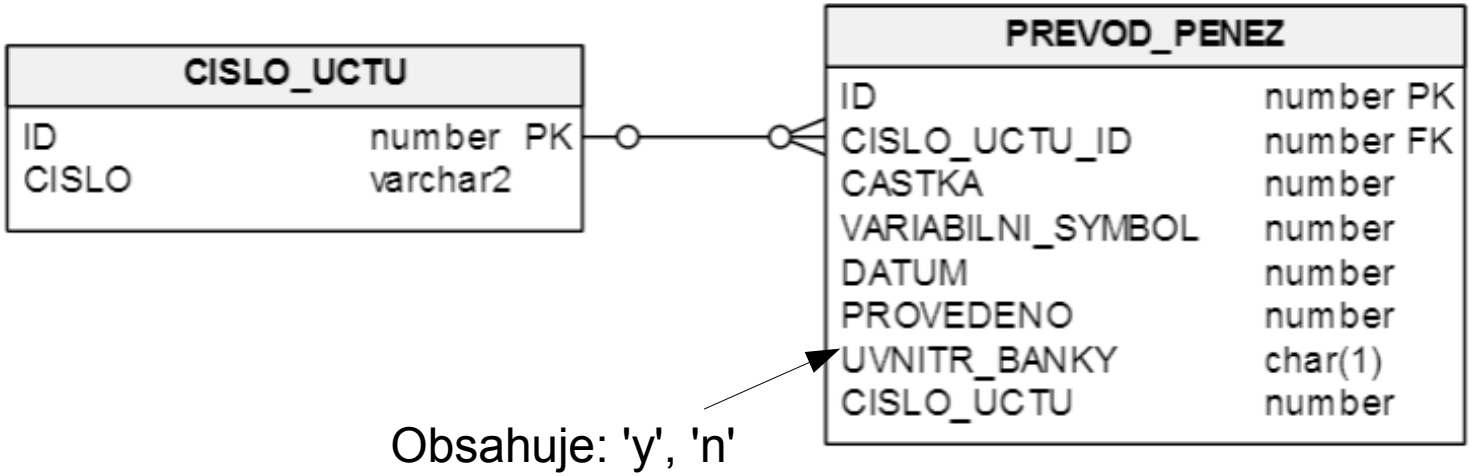
Transakce pro převod peněz I.

- Pro převod peněz je nutné udělat dva INSERT příkazy.
- Pro zjištění všech transakcí je nutné spojit tři tabulky.



Transakce pro převod peněz II.

- Pro převod peněz je nutné udělat pouze jeden INSERT příkaz.
- Pro zjištění všech transakcí se získají data jenom z jedné tabulky.



Alternativa: materializovaný pohled

- Existuje ještě jeden způsob jak zvýšit efektivitu a zachovat normalizovaný návrh databáze: materializovaný pohled.
- Mohli bychom mít normalizovanou databázi a vytvořit materializovaný pohled, který by obsahoval „denormalizovaný“ pohled.
- Poté bude efektivita operací následující:
  - Pro převod peněz je nutné udělat dva INSERT příkazy (+ režie s aktualizováním materializovaného pohledu).
  - Pro zjištění všech transakcí se získají data jenom z jedné tabulky (materializovaného pohledu).

Pořadí sloupců I.

- Data v řádku tabulky jsou uloženy v bloku sekvenčně a Oracle je hledá od prvního sloupce. Přičemž každý sloupec může mít variabilní délku. Co to znamená?

H = header, Ln = column length, Dn = column data

- Aby Oracle zpřístupnil data ve sloupci tři, musí začít ve sloupci jedna, přejít na sloupec dva a pak teprve může zpřístupnit data ve sloupci tři.
- To znamená, že data v počátečních sloupcích jsou zpřístupněny mnohem rychleji než data v koncových sloupcích, zejména pokud má tabulka hodně sloupců.
- Pokud má tabulka 250 sloupců, pak zpřístupnění posledního je 5x pomalejší než zpřístupnění prvního sloupce.

Pořadí sloupců II.

- Je best practice mít v prvních sloupcích v tabulce data, se kterými se pracuje nejčastěji.
- Také byste neměli získávat všechny sloupce z tabulky (`SELECT * FROM ...`), pokud skutečně všechny data nepotřebujete – protože se musí provést parsování těchto dat, server je musí načíst do operační paměti, musí se přenést ke klientovi, klient je musí načíst do operační paměti atd.
- Protože NULL hodnoty v koncových sloupcích nejsou uloženy, je best practice dávat na konec tabulky sloupce, ve kterých pravděpodobně budou NULL hodnoty.

Vhodný datový typ I.

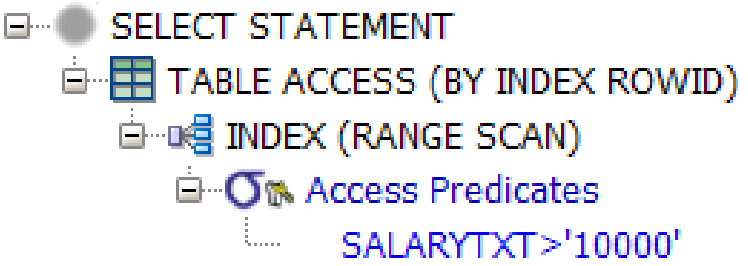
- Měli byste dbát na volbu vhodného datového typu sloupců.
- Při špatně zvoleném datovém typu:
  - Bude složitější kontrola integritních omezení.
  - Budete mít problémy při řazení.
  - Dá se lehce degradovat výkon SELECTů:
- Spusťte tento skript:

```
alter table hr.employees add (salarytxt varchar(10));
update hr.employees set salarytxt = salary;
create index hr.i_salarytxt on hr.employees(salarytxt);
```

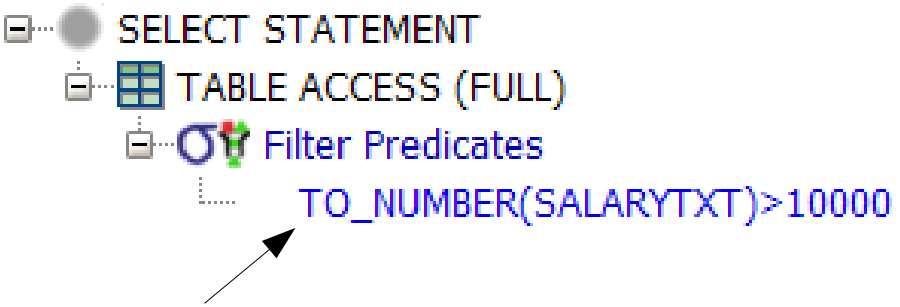
Vhodný datový typ II.

- Porovnejte Explain plan těchto SELECTů:

```
select * from hr.employees where salarytxt > '10000';
```




```
select * from hr.employees where salarytxt > 10000;
```



Implicitní konverze

Vhodný datový typ III.

- Pro zrušení sloupce `salarytxt` spusťte:

```
alter table hr.employees drop column salarytxt;
```